04459203 **Image available**
EMULATOR

PUB. NO.: 06-103103 [JP 6103103 A]
PUBLISHED: April 15, 1994 (19940415)
INVENTOR(s): SUGIYAMA SHIGEKI
APPLICANT(s): FUJIKURA LTD [000518] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 04-248297 [JP 92248297]
FILED: September 17, 1992 (19920917)
INTL CLASS: [5] G06F-011/26
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units)
JAPIO KEYWORD: R011 (LIQUID CRYSTALS); R131 (INFORMATION PROCESSING -- Microcomputers & Microprocessers)
JOURNAL: Section: P, Section No. 1770, Vol. 18, No. 376, Pg. 140, July 14, 1994 (19940714)

ABSTRACT

PURPOSE: To imitate the power source abnormal state of one computer by the other computer by imitating a function corresponding to the other computer side by an imulating means constituting of each function routine, defined by the same argument as a program for one computer.

CONSTITUTION: An emulator EM is constituted of a program group redefined so that various functions defined so as to operate in the hardware environment of a handy terminal operate in the hardware environment of a developing machine, prescribes the external specifications of each function by the same argument as a terminal exclusive function library, and also, each function inside redefines so as to emulate an operation of the handy on the developing machine. That is, the terminal exclusive function library is replaced with the emulator EM, debug DG2 of an application program is executed on the developing machine, down-load of an execution file Ef and actual machine debug on the handy terminal are omitted, and a low battery state is simulated.

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号

特開平6－103103

(43)公開日　平成６年(1994)４月15日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　11/26		8323－5B		

審査請求　未請求　請求項の数１(全 11 頁)

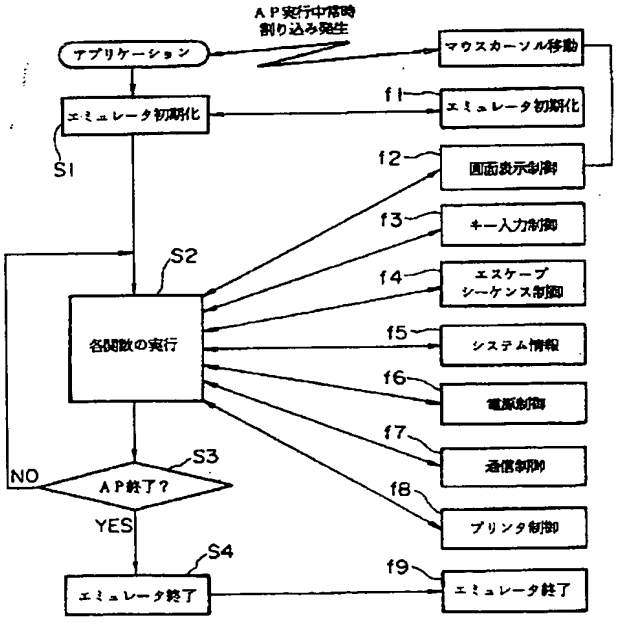
(21)出願番号　特願平4－248297

(22)出願日　平成４年(1992)９月17日

(71)出願人　000005186
株式会社フジクラ
東京都江東区木場１丁目５番１号
(72)発明者　杉山　茂樹
千葉県佐倉市六崎1440番地　藤倉電線株式会社佐倉工場内
(74)代理人　弁理士　志賀　正武

(54)【発明の名称】　エミュレータ

(57)【要約】　（修正有）
【目的】　ハンディターミナル１のローバッテリ状態を開発マシン上でシミュレートすることができるエミュレータを実現する。
【構成】　ハンディターミナル１用に作成したプログラムと同一の引数で定義された各関数ルーチンから構成されるエミュレータＥＭが、それぞれ開発マシン側のハードウェア構成に対応した各機能をエミュレートすると共に、このエミュレートされるハンディターミナル１のローバッテリ状態を、開発マシンにおける所定のキー操作によって模倣する。



1

【特許請求の範囲】

【請求項１】　互いにハードウェア構成が異なるコンピュータの内、いずれか一方のコンピュータ用に作成したプログラムを、他方のコンピュータ上で動作可能にするエミュレータにおいて、

前記プログラムがコールする複数のルーチンから形成されるライブラリであって、前記プログラムと同一の引数で定義された各関数ルーチンが、それぞれ前記他方のコンピュータ側のハードウェア構成に対応した各機能を模倣する模倣手段と、

所定のキー操作を検出する検出手段とを具備し、

前記模倣手段は、前記検出手段によって所定のキー操作がなされた場合に前記一方のコンピュータにおける電源異常状態を、前記他方のコンピュータで模倣することを特徴とするエミュレータ。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】この発明は、例えば、「ハンディターミナル」と呼ばれる携帯可能なコンピュータ用のアプリケーションプログラム開発に用いて好適なエミュレータに関する。

【０００２】

【従来の技術】近年、「ハンディターミナル」と呼ばれる携帯可能なコンピュータが各種実用化されている。この種のコンピュータは、ポータブルに構成されており、携行先で種々のデータ処理が可能になるため、様々な分野で使用されている。図７は、このようなハンディターミナル１の一構成例を示すブロック図である。

【０００３】この図において、１ａはハンディターミナル１の各部を制御するＣＰＵである。１ｂは、ＣＰＵ１ａの基本的な動作を制御するオペレーティングシステムプログラム（以下、これをＯＳと略す）が記憶されるＲＯＭである。１ｃは対応業務用のアプリケーションプログラムがロードされると共に、該プログラムのワークエリアとして各種レジスタ値が一時記憶されるＲＡＭである。１ｄはＬＣＤ（液晶表示素子）等から構成される表示回路であり、内部バスを介してＣＰＵ１ａから供給される各種データを表示する。１ｅはＬＣＤ（液晶表示素子）上に設けられた透明タッチパネル、あるいは本体パネルに配置されるテンキーやファンクションキー等から構成される操作子であり、それぞれ各操作に応じた操作子信号を発生する。

【０００４】１ｆは本体から着脱自在に構成されるメモリカードである。このメモリカード１ｆには、上述したアプリケーションプログラム、あるいは当該プログラムによって参照される各種データが記憶される。なお、メモリカード１ｆに記憶されるデータは、図示されていない上位コンピュータからダウンロードされるものであり、電池によってバックアップされる。１ｇは、例えば、モデム等から構成され、シリアルデータ通信を制御する通信制御回路である。このような構成によるハンディターミナル１は、周知のコンピュータと同様に動作する。つまり、電源投入後にＯＳが立上がると共に、このＯＳ上で対応業務用のアプリケーションプログラムがデータ処理を実行する。

2

【０００５】ところで、ハンディターミナル１で動作するアプリケーションプログラムは、「Ｃ言語」と呼ばれる構造化プログラミング言語で記述される場合が多い。図８は、「Ｃ言語」によるプログラム開発手順を示す図である。この図に示すように、プログラム開発は、コーディング、コンパイルおよびリンクの各作業からなり、実機デバッグを経て当該プログラムの動作が検証される。なお、このコーディング、コンパイルおよびリンクの各作業は、通常、開発マシン（例えば、パーソナルコンピュータ）上で行われる。

【０００６】コーディングにおいては、予め定められたシステム仕様に基づき、ソースプログラムをＣ言語で記述する。このソースプログラムは、ソースファイルｓｆとして開発マシン上に登録される。ソースファイルｓｆは、コンパイラＣＣの入力ファイルとなる。コンパイラＣＣでは、Ｃ言語で記述されたソースプログラムを語彙解析、構造解析および意味解析し、この結果をリロケータブルな中間コードで記述されたオブジェクトプログラムに変換する。このオブジジェクトプログラムは、開発マシン上でオブジェクトファイルｏｆとして登録される。

【０００７】リンカＬＫでは、オブジェクトファイルｏｆに対し、標準関数ライブラリＳＬと、ターミナル専用関数ライブラリＴＬとを結合させ、マシン語で記述された実行ファイルＥｆを生成する。ここで、標準関数ライブラリＳＬは、Ｃ言語において定義される各種の制御関数ルーチンプログラムから構成されている。また、ターミナル専用関数ライブラリＴＬは、ハンディターミナル１のハードウェア環境で動作するように定義された専用プログラム群から構成されるものである。

【０００８】すなわち、リンカＬＫでは、オブジェクトプログラムにおいてコールされる種々のルーチンプログラムや専用プログラムが、上述した各ライブラリＳＬ，ＴＬから引用され、これらをオブジェクトプログラムに結合させる。これにより、リロケータブルな中間コードで記述されたプログラムが実行形式に変換され、実行ファイルＥｆとして生成される。このようにして生成される実行形式のアプリケーションプログラムは、絶対アドレス上に展開可能な形態となる。

【０００９】次に、実行ファイルＥｆは、例えば、開発マシンから前述したメモリカード１ｆに書き込まれ、該メモリカード１ｆを介してＣＰＵ１ａにダウンロードＤＬされる。これにより、ハンディターミナル１上で実機デバッグＤＧ１が施される。実機デバッグＤＧ１においては、予め策定された検査項目に従ってアプリケーショ

ンプログラムの動作を検証する。この実機デバッグＤＧ１でバグが露見した場合には、ソースプログラムの対応箇所を修正する。そして、修正されたソースプログラムは、再びコンパイル、リンク作業を経て実行ファイル化されて実機デバッグＤＧ１が繰り返される。

【００１０】

【発明が解決しようとする課題】さて、上述したプログラム開発手順においては、ハンディターミナル１上でバグが露見する度毎に、当該バグに対応するソースプログラム部分を修正し、これを再度実行ファイル化してハンディターミナル１にダウンロードしなければならない。このため、実機デバッグＤＧ１には多大な工数が費やされ、結果的に開発コスト上昇を招致するという弊害がある。

【００１１】そこで、こうした弊害を解決するには、上述したコーディング、コンパイル、リンクおよびデバッグからなる一連の作業を全て開発マシン上で行い、かつ、デバッグ作業時には、特に、開発マシン上でハンディターミナル１の動作状態を全て把握できるようになることが要求される。これを換言すれば、開発マシン上でアプリケーションプログラムを実行し、ハンディターミナル１の動作をエミュレートできれば、上述した欠点が解消され、効率の良いプログラム開発が可能になる訳である。

【００１２】ところが、ハンディターミナル１と開発マシンとは、ほとんどの場合、ハードウェア環境が全く異なる。動作電源を例にとっても、ハンディターミナル１では、携行性が重視されるためにバッテリ（蓄電池）や電池等が用いられるのに対し、開発マシンでは、一般に商用電源が用いられる。ハンディーターミナル１本体に用いられる電源は主電池および副電池に分かれており（図示せず）、副電池は、主電池の交換時においてＲＡＭ１ｃの記憶内容が失われないようにするためのバックアップ電源である。さらに、メモリカード１ｆにも記憶内容をバックアップするために、電池が使用されている。これら電池を長時間使用すると、電源異常の状態、いわゆる、ローバッテリ状態となり、ハンディターミナル１における異常動作や動作停止の原因となる。これを防止するために、ハンディターミナルには、ローバッテリ状態を検出する関数が備えられ、この関数によりローバッテリ状態を検出し、例えば、その旨の表示を行うような処理がなされて、使用者に注意を促すようにしている。

【００１３】しかしながら、開発マシンに用いられる商用電源は、バッテリとは異なり、ローバッテリ状態になることはないため、開発マシン上では、ハンディターミナルのローバッテリ状態における処理を模倣することができない。このため、従来では、ハンディターミナル１にてローバッテリ状態となるのを待ち、ハンディターミナル１自身によってローバッテリ状態における処理を動作確認するという方法が採られている。この方法では、所望のときにローバッテリ状態を発生させることができないので、ハンディターミナル１用のプログラム、特に、ローバッテリ状態になって動作するプログラムの開発効率は極めて悪い、という問題があった。

【００１４】この発明は上述した事情に鑑みてなされたもので、ハンディターミナル１のローバッテリ状態を開発マシン上で模倣できるエミュレータを提供することを目的としている。

【００１５】

【課題を解決するための手段】この発明は、互いにハードウェア構成が異なるコンピュータの内、いずれか一方のコンピュータ用に作成したプログラムを、他方のコンピュータ上で動作可能にするエミュレータにおいて、前記プログラムがコールする複数のルーチンから形成されるライブラリであって、前記プログラムと同一の引数で定義された各関数ルーチンが、それぞれ前記他方のコンピュータ側のハードウェア構成に対応した各機能を模倣する模倣手段と、所定のキー操作を検出する検出手段とを具備し、前記模倣手段は、前記検出手段によって所定のキー操作がなされた場合に前記一方のコンピュータにおける電源異常状態を、前記他方のコンピュータで模倣することを特徴としている。

【００１６】

【作用】この発明によれば、互いにハードウェア構成が異なるコンピュータ間において、一方のコンピュータ用に作成されたプログラムを他方のコンピュータ上で動作する可能にする模倣手段が、検出手段による所定のキー操作によって一方のコンピュータにおける電源異常状態を、他方のコンピュータで模倣する。

【００１７】

【実施例】以下、図面を参照してこの発明の実施例について説明する。図１はこの発明の一実施例を適用したプログラム開発手順の概要を示す図である。この図において、図８に示す各部と共通する部分には、同一の符号を付し、その説明を省略する。図１に示す手順が図８に示した従来例と異なる点は、ターミナル専用関数ライブラリＴＬ（図８参照）を後述するエミュレータＥＭに置き換え、これにより、開発マシン（パーソナルコンピュータ）上でハンディターミナル１用に作成されたアプリケーションプログラムのデバッグＤＧ２を行うようにした点にある。すなわち、この実施例が意図するところは、エミュレータＥＭを用いたことにより、従来必要とされていた実行ファイルＥｆのダウンロードＤＬと、これに応じてなされるハンディターミナル１上の実機デバッグＤＧ１とを省略し、かつ、開発マシン上でハンディターミナル１のローバッテリ状態をシミュレートするようにした点にある。

【００１８】次に、図１のプログラム開発手順を実現するエミュレータＥＭの機能概要について説明する。ま

5

ず、エミュレータＥＭは、ハンディターミナル１のハードウェア環境で動作するよう定義された各種関数を、開発マシンのハードウェア環境で動作するように定義し直したプログラム群から構成されている。

【００１９】エミュレータＥＭの機能は、図２に示すように、ターミナル専用関数ライブラリＴＬ（図８参照）と同一である。つまり、このエミュレータＥＭでは、ターミナル専用関数ライブラリＴＬと同様の引数で各関数の外部仕様を規定し、かつ、各関数内部は、開発マシン上でハンディターミナル１の動作をエミュレートするよう定義し直している。

【００２０】図２において、ｆ１はエミュレータ初期化機能である。このエミュレータ初期化機能ｆ１は、ソースプログラムの先頭に「ｅｍｕ＿ｓｔａｒｔ（）」なる関数が記述されている場合、エミュレータＥＭの初期設定を行うものである。したがって、図１に示すように、オブジェクトファイルｏｆとエミュレータＥＭとをリンクさせる場合には、この「ｅｍｕ＿ｓｔａｒｔ（）」なる関数が必要になる。なお、この関数は、ターミナル専用関数ライブラリＴＬとリンクする際には、何等実行に影響されないものである。したがって、エミュレータ用に作成されたソースファイルｓｆは、そのままハンディターミナル１用の実行ファイルに変換可能になる。

【００２１】次に、ｆ２は画面表示制御機能である。この画面表示制御機能ｆ２では、ハンディターミナル１用に定義された各関数、例えば、表示モードの設定、カーソル移動、カーソル形状あるいは表示文字のアトリビュート設定等を行う各種の関数を、それぞれ開発マシンのハードウェア環境に対応して動作させる機能である。ｆ３はキー入力制御機能であり、ハンディターミナル１でなされるタッチパネル入力やテンキー入力を、開発マシン上におけるマウス入力やキー入力に置き換える機能である。ｆ４はエスケープシーケンス制御機能であり、カーソル位置指定、全画面消去あるいはスクロールアップ／ダウンなどを指定するコード出力関数を、開発マシンのハードウェア環境に対応して動作させる機能である。

【００２２】ｆ５は、メインメモリ容量、ドライバ登録情報、外字登録情報等のシステム情報を取得して出力する関数を開発マシンに対応させた機能である。ｆ６は主電源オン／オフ制御や、ローバッテリ状態検出などの動作を開発マシン上で表示させる電源制御機能である。ｆ７は、ハンディターミナル１用に定義されたシリアルデータ通信制御を、開発マシン上でエミュレートさせる通信制御機能である。ｆ８はプリンタ制御機能であり、ハンディターミナル１用に定義された印字フォント、改行ピッチなどを開発マシン上で制御するものである。ｆ９は、エミュレータ終了機能であり、ソースプログラムの最後、あるいは途中に「ｅｍｕ＿ｅｘｉｔ（ｎ）」なる関数が記述されている場合、エミュレータＥＭを終了させる機能である。

6

【００２３】次に、上記エミュレータＥＭを用いたプログラム開発手順と、この手順に基づき作成されたアプリケーションプログラムのエミュレート動作概要と、該エミュレート動作におけるマウス入力処理とについてそれぞれ説明する。アプリケーションプログラムを開発する際には、まず、コーディング段階でソースプログラムの先頭に「ｅｍｕ＿ｓｔａｒｔ（）」なる関数を記述し、かつ、該プログラムの最後に「ｅｍｕ＿ｅｘｉｔ（ｎ）」なる関数を記述しておく。そして、図１に示すように、ソースファイルｓｆをコンパイルｃｃにかけ、オブジェクトファイルｏｆを生成する。

【００２４】次に、このオブジェクトファイルｏｆに対して標準関数ライブラリＳＬおよびエミュレータＥＭをリンクさせる。これにより、ソースプログラムでコールされるハンディターミナル固有の各種関数が開発マシン上で動作する形で実行ファイル化される。実行ファイル化されたアプリケーションプログラムは、開発マシンの主メモリ上にロードされてハンディターミナル１の動作をエミュレートする。

【００２５】　アプリケーションプログラムのエミュレート動作

上記手順により作成されたアプリケーションプログラムが起動すると、開発マシンの処理は、図３に示すステップＳ１に進む。ステップＳ１では、プログラム先頭に定義された「ｅｍｕ＿ｓｔａｒｔ（）」なる関数に基づき、上述したエミュレータ初期化機能ｆ１が初期設定を行う。この初期設定とは、ハンディターミナル１の起動状態をエミュレートするものであり、例えば、バックライトのオン／オフ状態、液晶パネルのコントラスト状態あるいはスピーカ音量などを設定するものである。

【００２６】こうして開発マシン上でハンディターミナル１の初期状態がエミュレートされると、例えば、図４に示すように、開発マシンのディスプレイＤＳＰにハンディターミナル１の表示画面ＴＤが表示される。ここで、同図（イ）は、エミュレータ初期化機能ｆ１に基づきハンディターミナル１の初期化状態がエミュレートされた表示例である。一方、同図（ロ）は、ハンディターミナル１の表示画面ＴＤである。同図（イ）に示すように、ディスプレイＤＳＰは、表示エリアＥ１、Ｅ２、Ｅ３に３分割されており、この内の表示エリアＥ１には表示画面ＴＤと同一の表示がなされる。さらに、表示エリアＥ２には、ハンディターミナル１のタッチパネルの分割状態が表示され、表示エリアＥ３には、ハンディターミナル１の動作状態が表示される。

【００２７】次いで、このような初期設定がなされると、開発マシンの処理はステップＳ２に進む。ステップＳ２では、与えられたイベントに応じてアプリケーションプログラムが各関数を実行する。このステップＳ２においては、ハンディターミナル１のタッチパネル入力をマウス入力するようキー入力制御機能ｆ３が動作してお

り、これに対応してマウスカーソルを移動させる割込処理が実行される。

【００２８】したがって、図４（イ）に示すように、実際のタッチパネルをエミュレートした表示エリアＥ１上の所定位置をクリックしてマウス入力を行なうと、この入力イベントに対応したアプリケーションプログラムを実行し、該プログラム中で定義された関数が実行される。これにより、ハンディターミナル１の動作が開発マシン上でエミュレートされる。例えば、マウスカーソルＭＣが、図４に示すように、表示エリアＥ１上の「印刷」の表示に置かれ、マウスクリックがなされると、該「印刷」処理が該開発マシン上でエミュレートされる。

【００２９】このように各種処理がエミュレートされ、アプリケーションプログラム完了の旨を表わす入力がなされると、ステップＳ３の判断結果が「ＹＥＳ」となり、該プログラムの処理がステップＳ４に進む。ステップＳ４では、プログラム最後に定義された「ｅｍｕ＿ｅｘｉｔ（ｎ）」なる関数に基づき、エミュレート動作を終了する。

【００３０】次に、図５および図６を参照し、上記ステップＳ２においてなされるローバッテリ状態発生処理について説明する。なお、この動作説明においては、前述したキー入力制御機能ｆ３が起動されてキー入力可能状態にあり、かつ、この機能ｆ３がキーバッファにセットされた文字列を、後述するローバッテリ状態発生ルーチンに引き渡すものとする。ここで、ステップＳ２においてアプリケーションプログラムがキー入力状態にある場合、図５に示すローバッテリ状態発生ルーチンが、割込処理によって所定周期毎に起動され、キー入力状態をつねにチェックしている。

【００３１】まず、ステップＳＰ１では、現時点において、キーバッファにデータが有るか否かが判別される。データが存在しないときには、すなわち、なんらキー操作がなされていない場合には、判別結果が「ＮＯ」となって、このルーチンにおける以後の処理が実行されることなく、処理は直ちに終了する一方、データが存在するときには判別結果が「ＹＥＳ」となって、処理はステップＳＰ２に進み、キーバッファ内のデータがスキャンされ、さらに、ステップＳＰ３に進んで、スキャンされたデータが、ローバッテリ状態を発生すべき旨のキー操作に相当するものであるか否かが判別される。

【００３２】ここで、ローバッテリ状態を発生すべき旨のキー操作とは、予め決められた特定のキー操作方法をいうものであり、この実施例では、例えば、「ＣＴＲＬ」＋「Ｏ」、「ＣＴＲＬ」＋「Ｂ」、「ＣＴＲＬ」＋「Ｌ」のいずれかに相当している。これらキー操作のうち、「ＣＴＲＬ」＋「Ｏ」は、ハンディターミナル１における主電池のローバッテリ状態をエミュレートする場合に用いられ、「ＣＴＲＬ」＋「Ｌ」は、同副電池のローバッテリ状態をエミュレートする場合に用いられ、また、「ＣＴＲＬ」＋「Ｂ」は、メモリカード１ｆのローバッテリ状態をエミュレートする場合に用いられる。なお、「ＣＴＲＬ」＋「Ｏ」とは、「ＣＴＲＬ」（コントロール）キーと「Ｏ」キーとを同時に押下する操作を意味しており、これらの操作は、ハンディターミナル１側ではサポートされないものが用いられる。

【００３３】ステップＳＰ３の判別結果が「ＮＯ」であるならば、処理はステップＳＰ６に進み、ステップＳＰ２においてスキャンしたデータをキーバッファに戻し、このデータを他の処理にて用いるべくこのルーチンは終了する一方、判別結果が「ＹＥＳ」であるならば、処理はステップＳＰ４に進み、状態フラグＬＢ１～ＬＢ３をキー操作に対応してそれぞれオンにする。すなわち、押下されたキーが「ＣＴＲＬ」＋「Ｏ」であるならば、状態フラグＬＢ１がオンとなり、「ＣＴＲＬ」＋「Ｌ」であるならば、状態フラグＬＢ２がオンとなり、「ＣＴＲＬ」＋「Ｂ」であるならば、状態フラグＬＢ３がオンとなる。

【００３４】そして、図４に示す表示エリアＥ３の「電源状態」の状態表示が「正常」からキー操作に対応する「ＬＢ１」、「ＬＢ２」、「ＬＢ３」の内のいずれかに変更され、ローバッテリ状態をエミュレートしている旨の表示がなされて、このルーチンは終了する。このように、ローバッテリ状態発生ルーチンでは、図３におけるステップＳ２のキー入力可能状態においてなされた所定のキー操作に応じて、状態フラグＬＢ１～ＬＢ３のいずれかがオンとなり、これに対応して、表示エリアＥ３に該状態フラグの状態が表示される。

【００３５】次に、図６を参照し、上記ルーチンで設定された状態フラグＬＢ１～ＬＢ３を用いる関数例について説明する。図６に示すルーチンはローバッテリ状態取得関数の一例であり、ステップＳＢ１において、図５に示すルーチンによって設定された状態フラグＬＢ１～ＬＢ３が読み出され、ステップＳＢ２において、この読み出したフラグの内容が操作者に返される。このような関数を電源制御機能ｆ６（図３参照）が、開発マシンに対応させて動作させることによって、ローバッテリ状態検出などの動作を開発マシン上で表示することができる。これによって、従来、直接的に動作状態を検証できなかったり、目視で確認することができなかった、ローバッテリ状態におけるハンディターミナル１の動作が開発マシン上で一目瞭然となる。

【００３６】このように、エミュレータＥＭは、ハンディターミナル１のハードウェア環境に対応させた専用関数ライブラリＴＬと同一の引数を備え、かつ各専用関数の機能ｆ１～ｆ９を開発マシンのハードウェア環境に対応させている。このため、ソースプログラムにエミュレータＥＭをリンクして、生成した実行ファイルＥｆを開発マシンにロードすれば、該開発マシン上でハンディターミナル１の動作がエミュレートされることになる。さ

らに、このエミュレート動作では、ハンディターミナル１の動作状態がディスプレイＤＳＰに逐一表示されるたので、アプリケーションプログラムのデバッグ作業が極めて効率良く行える。

【００３７】また、上記実施例によれば、エミュレータ用に開発したソースプログラムがそのままハンディターミナル用のソースファイルとなり、完全互換性を備えるので、プログラム開発がすべて同一のマシン上で行なうことが可能になる。この結果、従来必要とされる実行ファイルＥｆのダウンロードＤＬと、これに応じてなされる実機デバッグＤＧ１とが省略することができる。さらに、開発マシンでは、通常では有り得ないローバッテリ状態を、所定の操作でいつでも、エミュレートすることが可能となる。これらによって、極めて効率の良いプログラム開発が可能となる。

【００３８】

【発明の効果】以上説明したように、この発明によれば、一方のコンピュータ用に作成したプログラムと同一の引数で定義された各関数ルーチンから構成される模倣手段が、他方のコンピュータ側のハードウェア構成に対応した各機能を模倣するので、一方のコンピュータにおける電源異常状態を、他方のコンピュータで模倣することができる。

【図面の簡単な説明】

【図１】　この発明の一実施例を適用したプログラム開発手順の概要を示す図である。

【図２】　同実施例におけるエミュレータＥＭの機能構成を示す図である。

【図３】　同実施例により作成されたアプリケーションプログラムの概略動作を示すフローチャートである。

【図４】　同実施例における表示エリアＥ１，Ｅ２，Ｅ３の表示例を示す図である。

【図５】　同実施例におけるローバッテリ状態発生ルーチンの動作を示すフローチャートである。

【図６】　同実施例におけるローバッテリ状態取得関数の動作を示すフローチャートである。

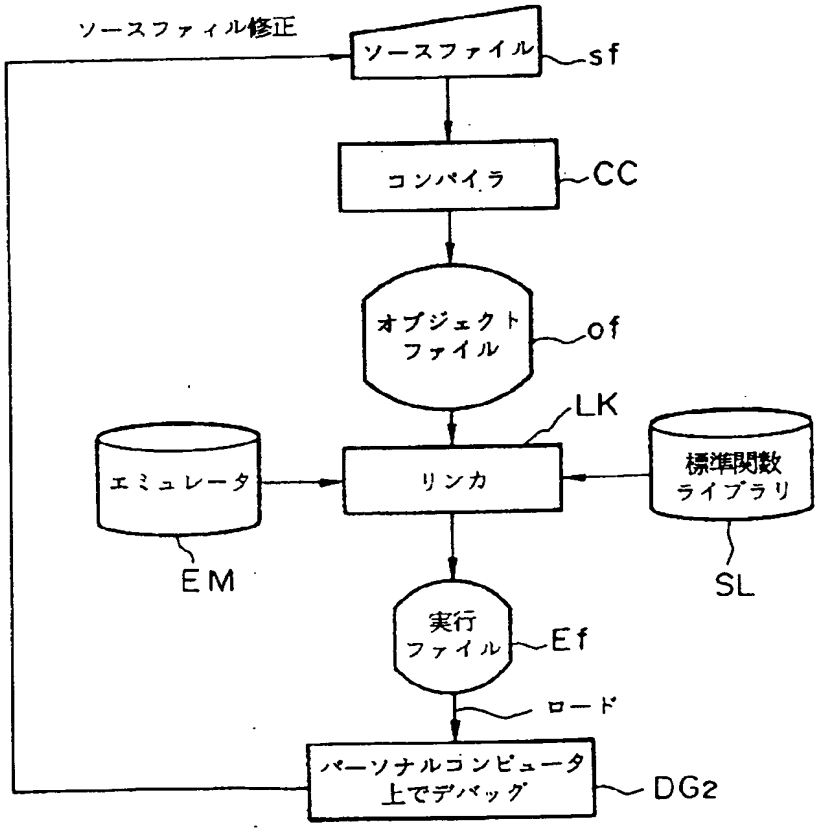
【図７】　ハンディターミナル１の一構成例を示すブロック図である。

【図８】　従来のプログラム開発手順を説明するための図である。

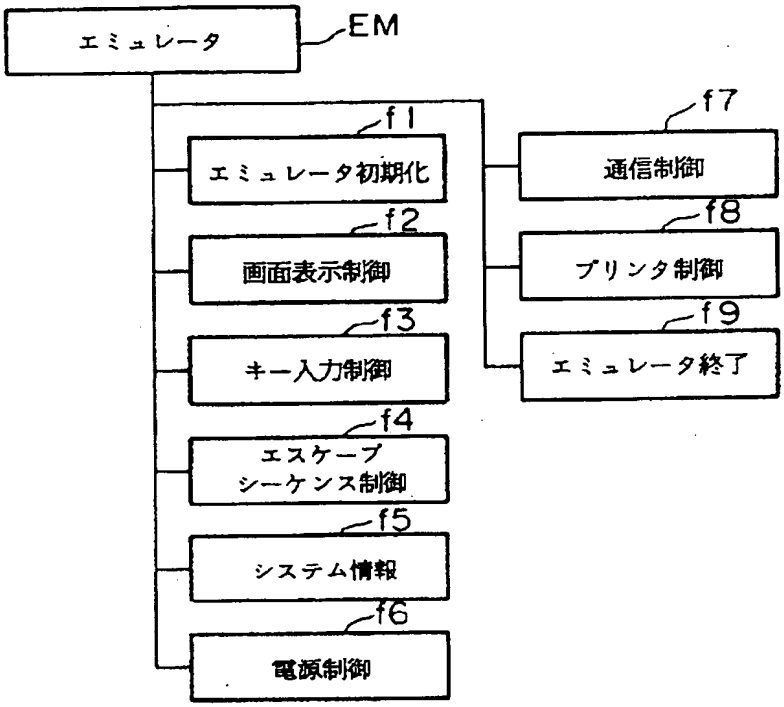
【符号の説明】

ｓｆ…ソースファイル、ｃｃ…コンパイル、ｏｆ…オブジェクトファイル、ＬＫ…リンカ、ＥＭ…エミュレータ（模倣手段）、　ＳＬ…標準関数ライブラリ、Ｅｆ…実行ファイル、ｆ３…キー入力制御機能、ｆ６…電源制御機能。

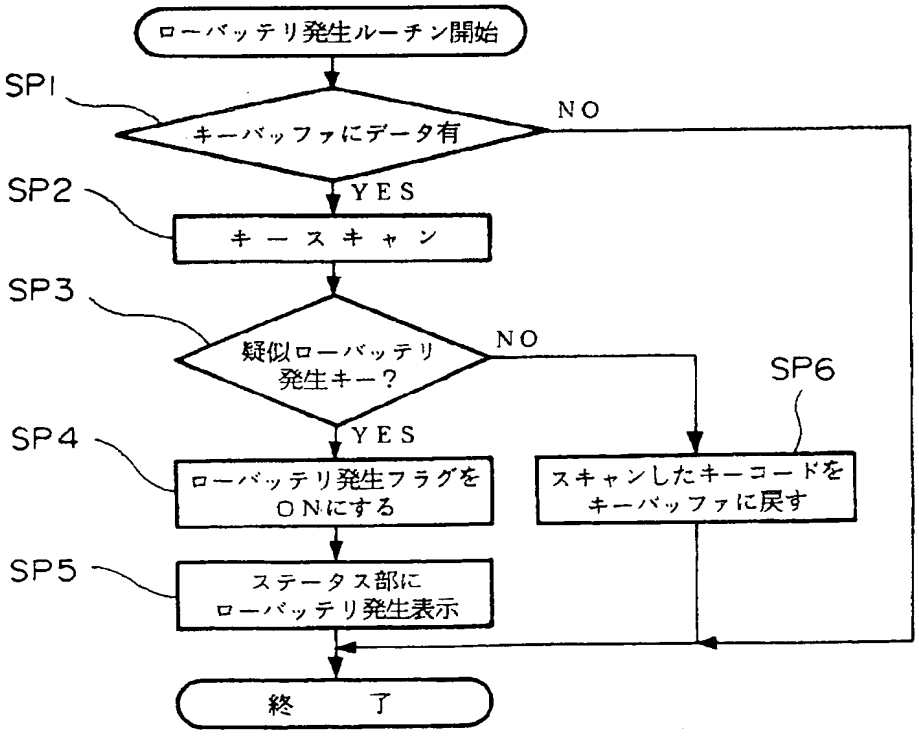
【図１】



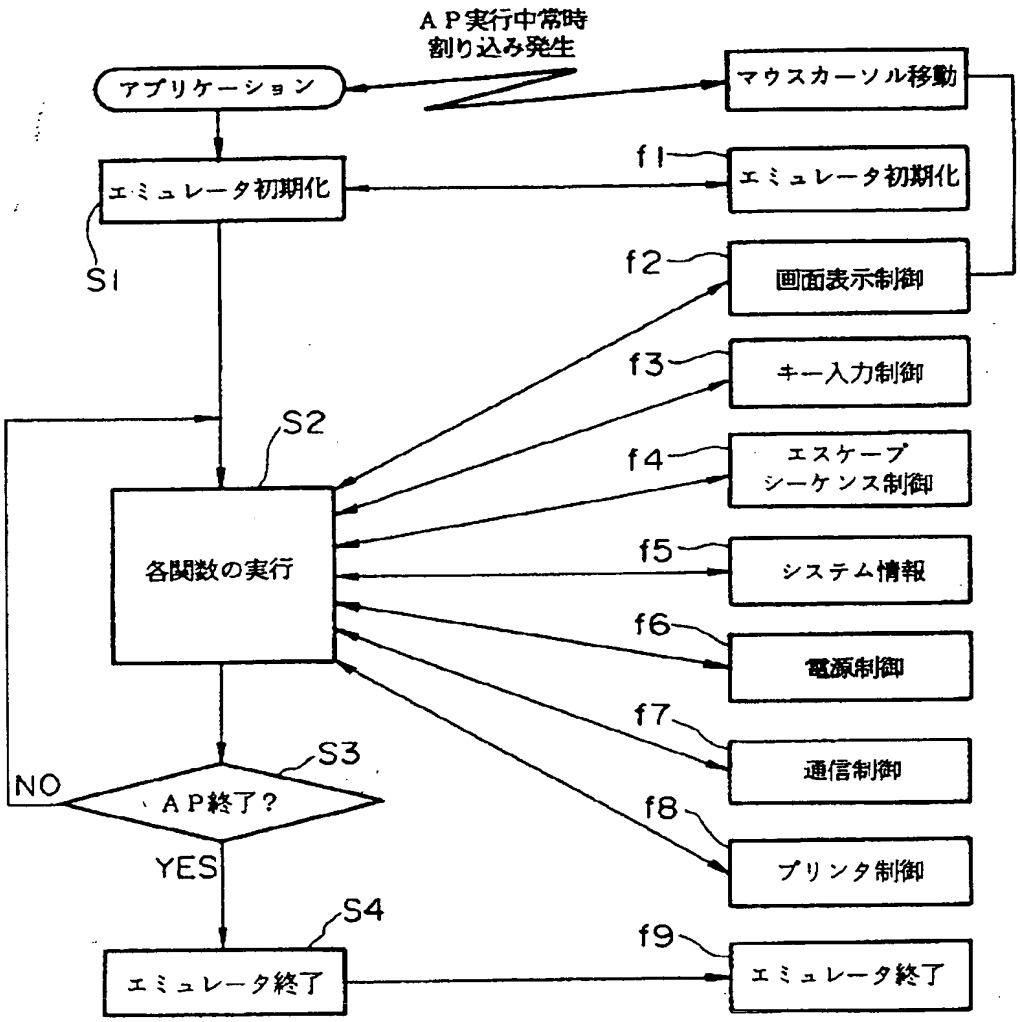
【図２】


エミュレータ
EM
f1
エミュレータ初期化
f2
画面表示制御
f3
キー入力制御
f4
エスケープ
シーケンス制御
f5
システム情報
f6
電源制御
f7
通信制御
f8
プリンタ制御
f9
エミュレータ終了

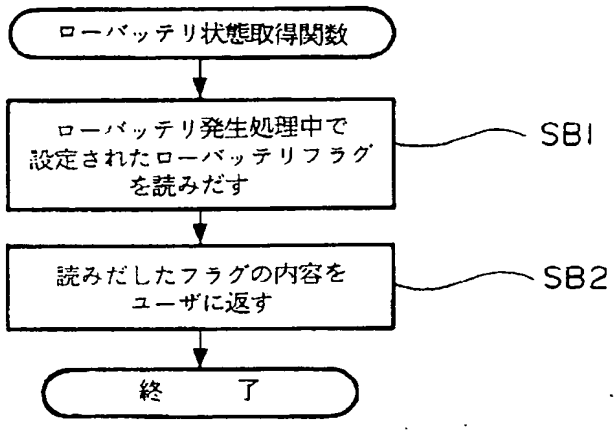
【図５】


ローバッテリ発生ルーチン開始
SP1
キーバッファにデータ有
NO
YES
SP2
キースキャン
SP3
疑似ローバッテリ
発生キー？
NO
YES
SP6
SP4
ローバッテリ発生フラグを
ONにする
スキャンしたキーコードを
キーバッファに戻す
SP5
ステータス部に
ローバッテリ発生表示
終　了

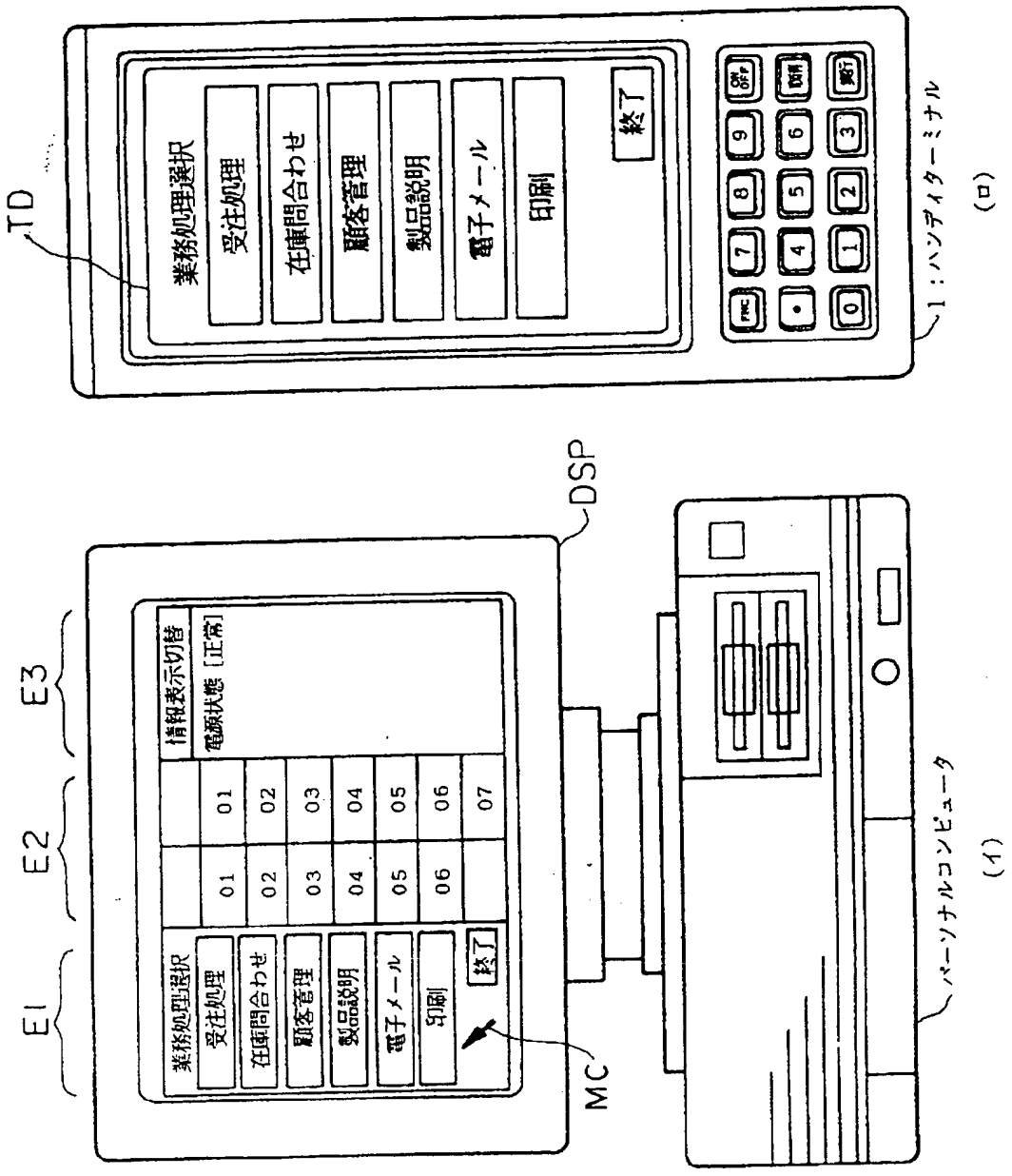
【図３】


ＡＰ実行中常時
割り込み発生
アプリケーション
マウスカーソル移動
エミュレータ初期化
f1
エミュレータ初期化
S1
f2
画面表示制御
f3
キー入力制御
S2
f4
エスケープ
シーケンス制御
各関数の実行
f5
システム情報
f6
電源制御
f7
通信制御
S3
NO
ＡＰ終了？
f8
プリンタ制御
YES
S4
f9
エミュレータ終了
エミュレータ終了

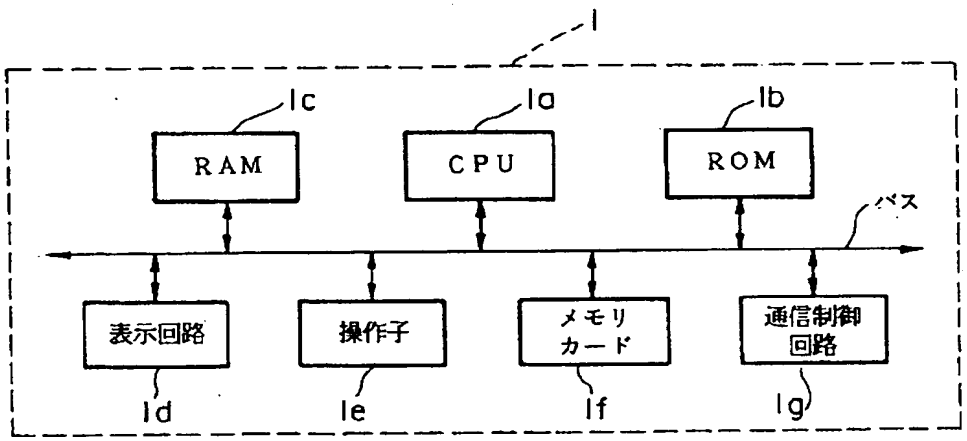
【図６】


ローバッテリ状態取得関数
ローバッテリ発生処理中で
設定されたローバッテリフラグ
を読みだす
SB1
読みだしたフラグの内容を
ユーザに返す
SB2
終　了

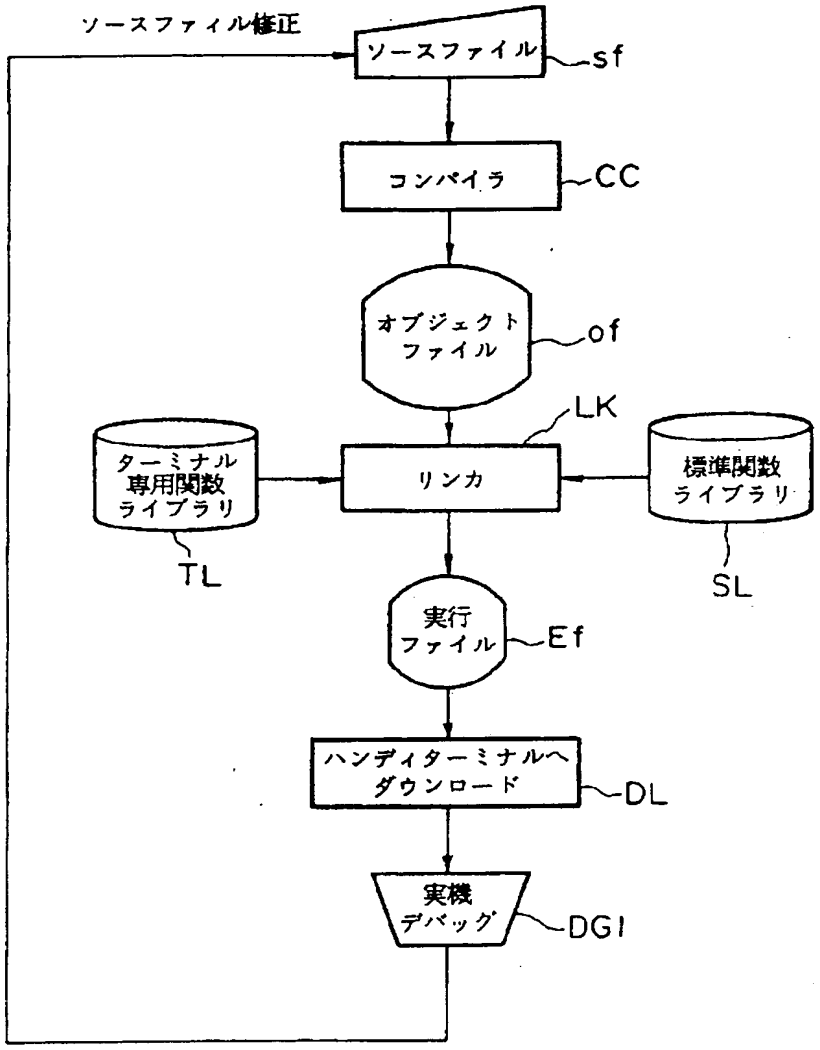
【図4】


TD
業務処理選択
受注処理
在庫問合わせ
顧客管理
製品説明
電子メール
印刷
終了
1：ハンディターミナル
(ロ)
E1
E2
E3
DSP
MC
業務処理選択
受注処理
在庫問合わせ
顧客管理
製品説明
電子メール
印刷
終了
情報表示切替
電源状態［正常］
パーソナルコンピュータ
(イ)

【図７】



【図８】


ソースファイル修正
ソースファイル
sf
コンパイラ
CC
オブジェクト
ファイル
of
LK
ターミナル
専用関数
ライブラリ
TL
リンカ
標準関数
ライブラリ
SL
実行
ファイル
Ef
ハンディターミナルへ
ダウンロード
DL
実機
デバッグ
DGI